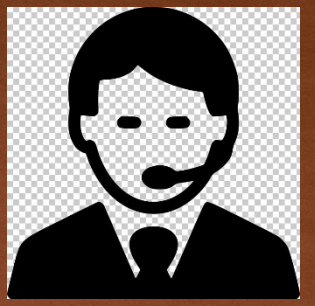
Roles of an OS



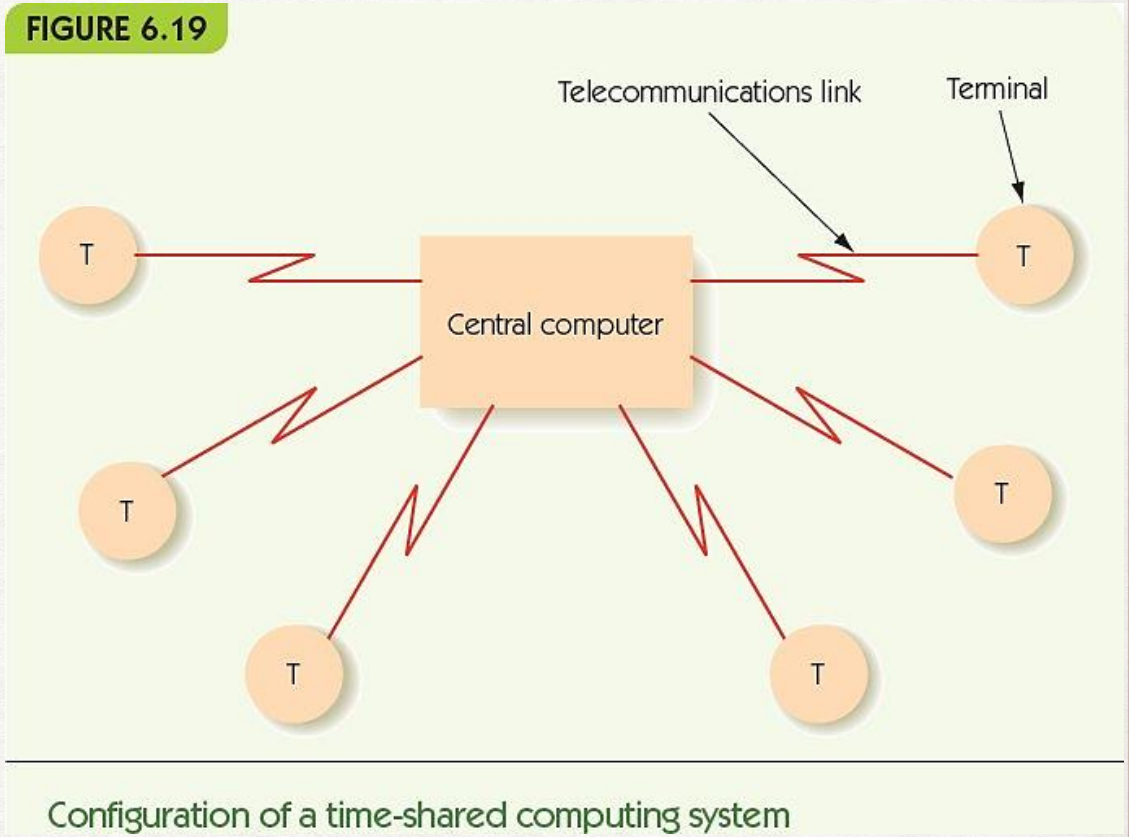
1. User Interface Management (a receptionist)
2. **Program Scheduling and Activation (a dispatcher)**
3. Efficient Resource Allocation (an efficiency expert)
4. Deadlock Detection and Error Detection (a traffic officer)
5. Control of access to the system and data (a security guard)

Generations of an OS : A Recap

Generation	Dates	Characteristics
First	1945 - 1955	**Naked Machines – no OS** Programmers operated the machine themselves
Second	1955 – 1965	**Batch Operating Systems** Several programs were grouped into a “batch” An operator loaded this batch of programs onto an input tape which was then fed to a main computer that executed them sequentially. Improved system utilization from the single user scenario.
Third	1965 – 1985	**Multiprogrammed OS** The OS keeps several programs in memory. If the currently running program pauses for I/O; the OS picks up one of the other “ready” process from a queue. Improves processor utilization by keeping the processor “busy” most of the time.
	1970s	**Time-sharing OS** Interactive use of a central computer system. Users sit at terminals (“dumb”) and are connected to the central computer via communication links
Fourth	1985 - Present	
Fifth	Future OSs	

Time- Shared OS (1970s)

- Multiprogrammed OS, but users are on the system interactively.
- Users need an illusion of sole access.
  - OS sets up **“virtual machines”** on which other programs run.
- Allocate run time in ***time slices*** - each program runs until it needs an I/O OR its time runs out.



OPERATING SYSTEM AS A DISPATCHER / SCHEDULER

- The OS needs to ensure that each process gets a **fair share** of processor time, and processes do not unnecessarily hog the processor.
  - In order to do so, the OS creates ***time slices***. A process only runs for as long as the time slice.
  - In practice, these are typically 10-20 milliseconds long.
  - Conceptually, we may think that the OS runs one "virtual machine" per time slice.

- The basic idea in a time-sharing system is to service many users in a circular, round-robin fashion, giving each one a small amount of time and then moving on to the next.
  - OS schedules a process, loads it, runs it for a while, puts it away, loads it again,
    - Known as "context switching"

Exercise : Time Sharing

Four processes to be executed on a single processor system arrive at time 0 in the order A, B, C, D. Their CPU time requirements are 4, 1, 8, 2 time slices respectively.

1. How many time slices are needed for completion of A (assuming processes are serviced one after the other in the order in which they arrive)?

   Answer : 10 time slices

Exercise：Time Sharing

Four processes to be executed on a single processor system arrive at time 0 in the order A, B, C, D. Their CPU time requirements are 4, 1, 8, 2 time slices respectively.

1. How many time slices are needed for completion of A (assuming processes are serviced one after the other in the order in which they arrive)?
2. If the time slice is 10 ms, how long will it take for process C to complete?

   Ans: 150ms

Exercise：Time Sharing

Four processes to be executed on a single processor system arrive at time 0 in the order A, B, C, D. Their CPU time requirements are 4, 1, 8, 2 time slices respectively.

1. How many time slices are needed for completion of A (assuming processes are serviced one after the other in the order in which they arrive)?
2. If the time slice is 10 ms, how long will it take for process C to complete?
3. If the OS runs for one time slice at the beginning and between each process (to save and load the context of processes), how long will it take for each process to complete?

   A: 20 * 10 = 200ms

   B: 4*10 = 40ms

   C: 26*10 = 260ms

   D: 14*10 = 140ms

Exercise：Time Sharing

Four processes to be executed on a single processor system arrive at time 0 in the order A, B, C, D. Their CPU time requirements are 4, 1, 8, 2 time slices respectively.

1. How many time slices are needed for completion of A (assuming processes are serviced one after the other in the order in which they arrive)?
2. If the time slice is 10 ms, how long will it take for process C to complete?
3. If the OS runs for one time slice at the beginning and between each process (to save and load the context of processes), how long will it take for each process to complete?
4. What is the efficiency of the processor?

   Ans: 50％

What did we learn?

The performance of time slicing policy is heavily dependent on the size/duration of the time slice.

- When the time slice is very large, it affects the other processes in the queue.
- Too short time slice causes too many process/context switches and reduces CPU efficiency.

$$\text{CPU efficiency (\%)} = \frac{\text{time slice}}{(\text{time slice} + \text{context switch time})} \times 100$$

The challenges of a multi programmed OS : OS as a dispatcher

Scheduling Considerations:

Turnaround time:

How much time does a program need to complete?

E.g., weather forecasts, financial data processing

Response Time

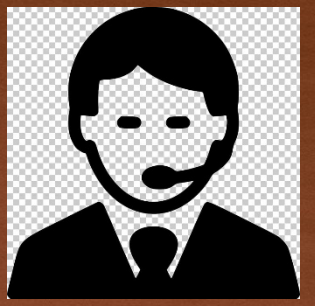
How often must a program process input?

E.g., follow mouse movement, respond to data traffic arriving, send / play audio.

Throughput

Number of processes completed per unit time.

Roles of an OS



1. User Interface Management (a receptionist)
2. Program Scheduling and Activation (a dispatcher)
3. **Efficient Resource Allocation (an efficiency expert)**
4. Deadlock Detection and Error Detection (a traffic officer)
5. Control of access to the system and data (a security guard)

OPERATING SYSTEM AS AN EFFICIENCY EXPERT

- Efficient Management of other resources
- Memory Management
  - **Memory spaces** of the different processes need to be kept separate.
    - If the program attempts to access a memory location, it is not allowed to or attempts to access a memory location in a way it is not allowed to (e.g., trying to write to a read-only memory location), a **segmentation fault** occurs.
  - OS also needs to allocate and reallocate memory as needed by the processes.

OPERATING SYSTEM AS AN EFFICIENCY EXPERT

- Efficient Management of other resources

- Device management
  - The OS needs to manage program access to external devices such as keyboard, mouse, network and printers.

- File management
  - The **file system** is the part of the OS which looks after storing and retrieving files from storage devices such as disk drives.

SOME OF THE IMPORTANT FILE INFORMATION

The file system needs to store information about each file (think about why each of these items is important)

- the file name
- the file size
- the time the file was most recently modified
- who owns the file
- where the file data is stored on the disk device
- who is allowed to do what to this file

All of this information must be stored on the disk too.

Roles of an OS

1. User Interface Management (a receptionist)
2. Program Scheduling and Activation (a dispatcher)
3. Efficient Resource Allocation (an efficiency expert)
4. **Deadlock Detection and Error Detection (a traffic officer)**
5. Control of access to the system and data (a security guard)

The challenges of a multi programmed OS : OS as a traffic officer

If two programs A and B both need to write to the same file, they should do so one after the other, not at the same time.

The OS typically achieves this by giving the respective program a *lock* on the file.

This is not a perfect solution, however:

- Say A needs to write to file X and Y and gets a lock for X, but needs to wait until B has finished writing to Y, which is locked for B.
- If B cannot finish writing to Y, however, because B in turn is also trying to write to X (which is locked for use by A), then neither A nor B can proceed:
  - We have a ***deadlock***.
    - An OS must detect and resolve such deadlock events.

The challenges of a multi programmed OS : OS as a traffic officer

A **deadlock** occurs when multiple programs are requesting the resources that each one is currently using.

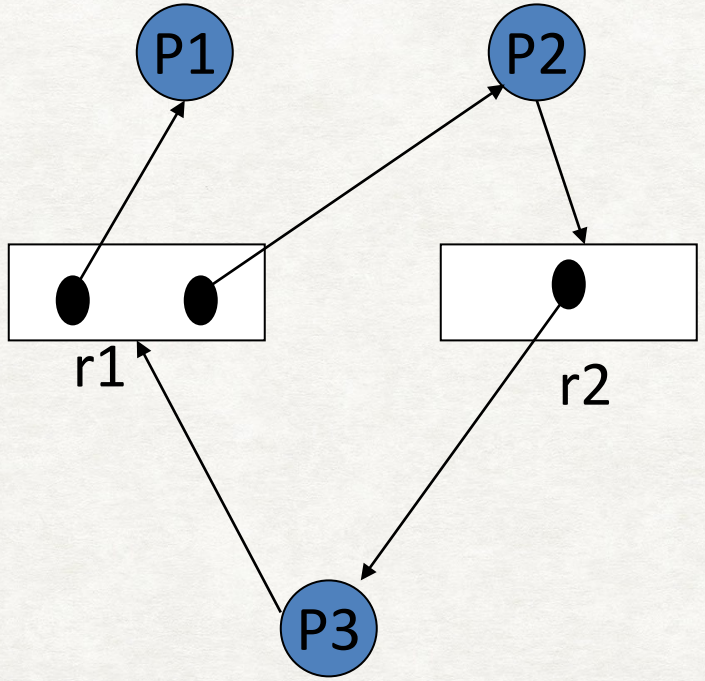
Safe use of resources: ensure that the computer doesn't get stuck in deadlock

Deadlock prevention: if you can't get all resources, release all you have and try again later

Deadlock recovery: Allows deadlocks to form; then finds and breaks them.

Exercise：Deadlock Detection

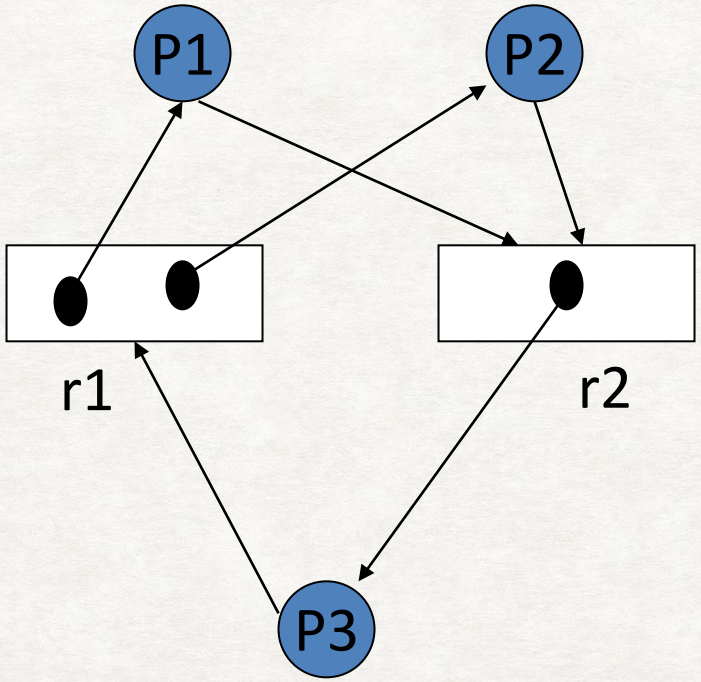
A system has 3 processes and 2 types of resources r1 and r2. There are 2 units of r1 and 1 unit of r2. Process 1 has locked one unit of r1. Process 2 has locked one unit of r1 and is waiting for r2 which is locked by process 3. Process 3 is waiting for r1. Is there a deadlock in the system?



No deadlock!

Exercise：Deadlock Detection

A system has 3 processes and 2 types of resources r1 and r2. There are 2 units of r1 and 1 unit of r2. Process 1 has locked one unit of r1 and is waiting for r2. Process 2 has locked one unit of r1 and is waiting for r2 which is locked by process 3. Process 3 is waiting for r1. Is there a deadlock in the system?



Deadlocked!